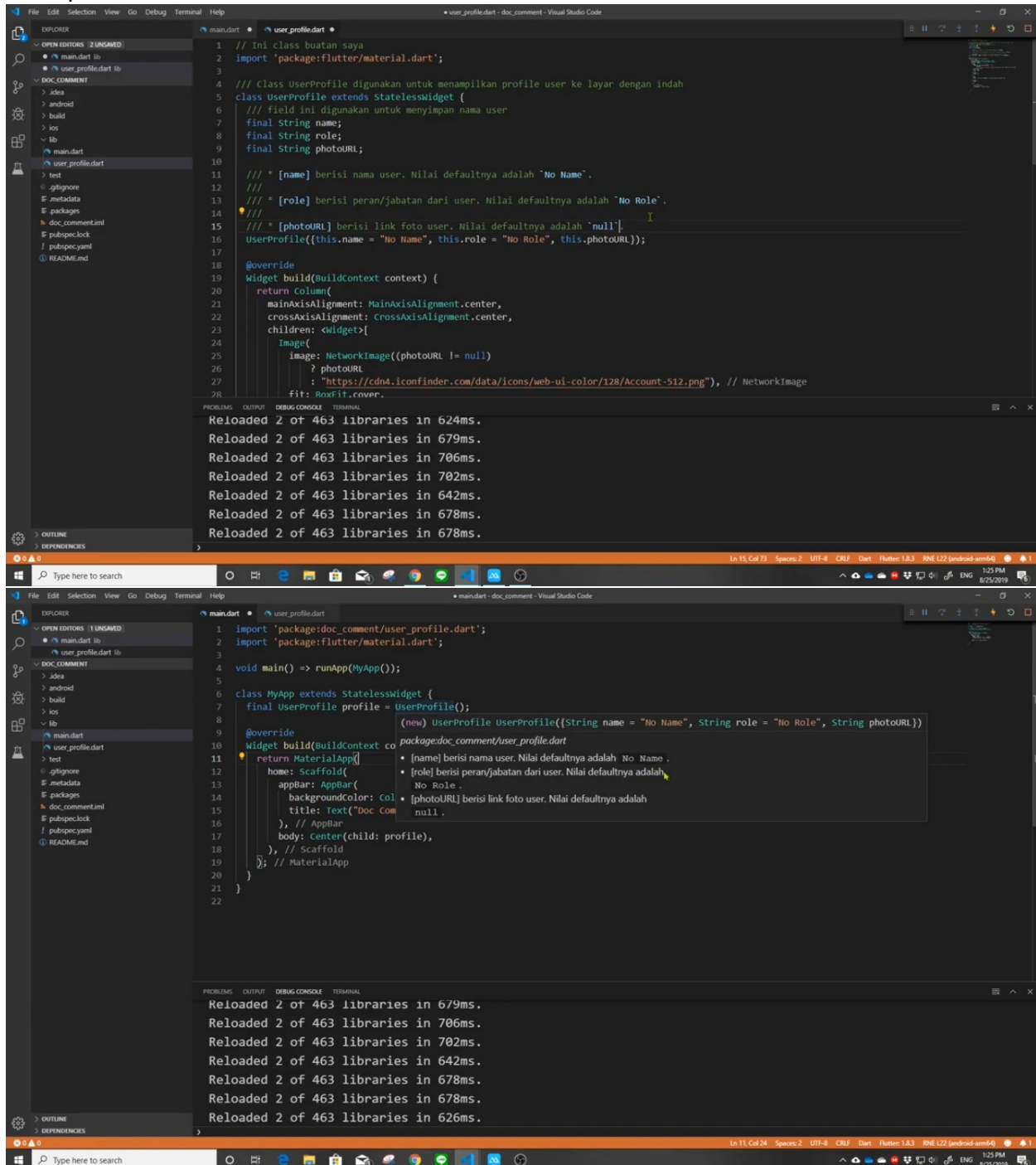
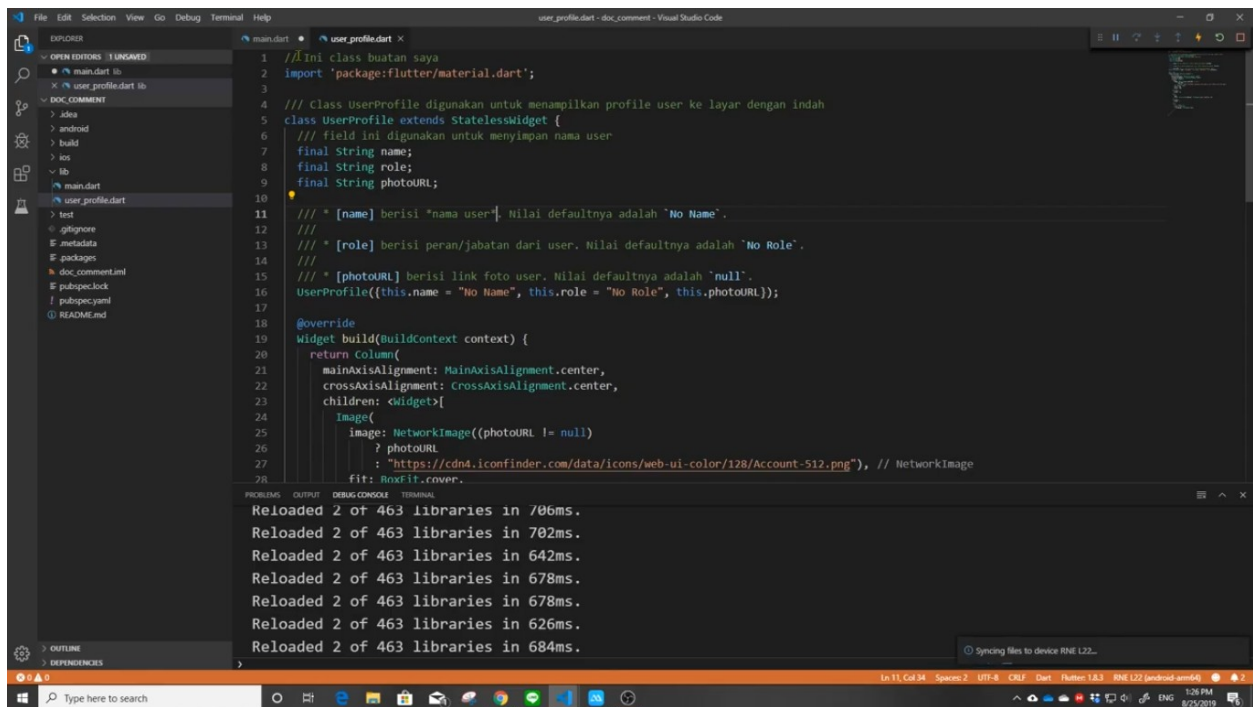


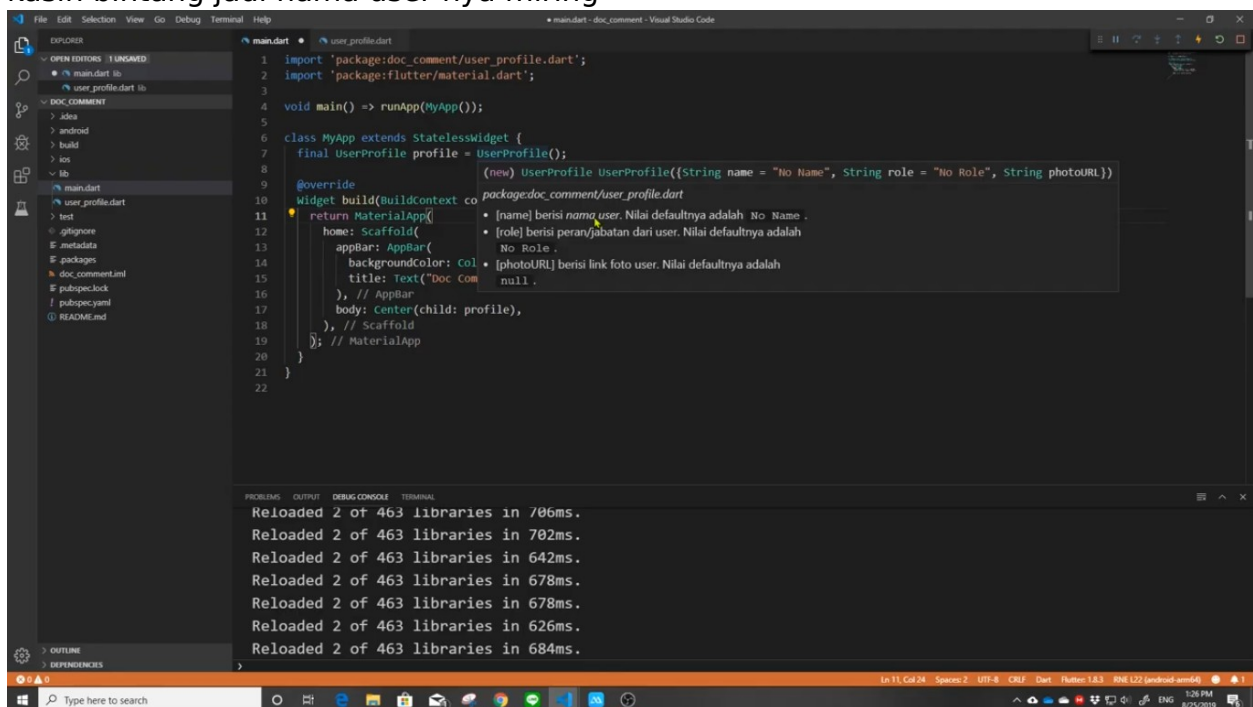
Terus untuk nilai bisa kita berikan seperti ini
Bactip diatas tab



Dan juga bisa berikan effect italic contoh nya nama users



kasih bintang jadi nama user nya miring



Dan juga bisa kasih bold
 Cara nya double bintang

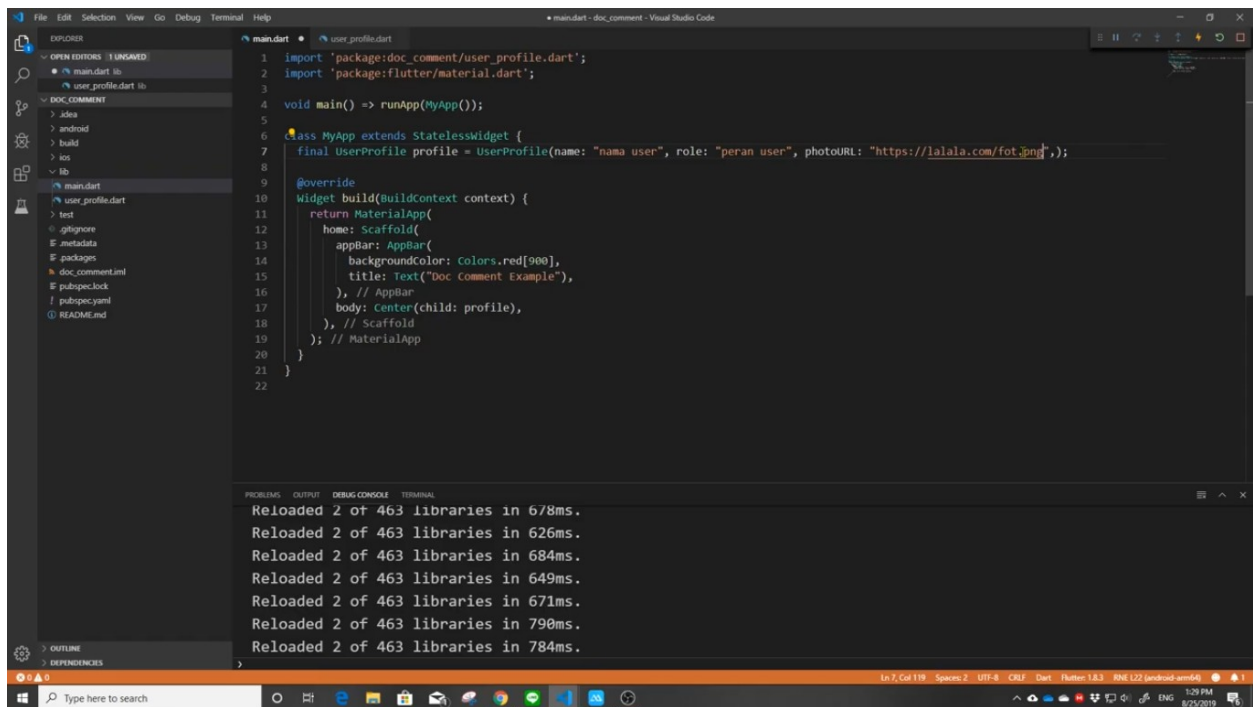
```
1 // Ini class buatan saya
2 import 'package:flutter/material.dart';
3
4 /// Class UserProfile digunakan untuk menampilkan profile user ke layar dengan indah
5 class UserProfile extends StatelessWidget {
6   /// field ini digunakan untuk menyimpan nama user
7   final String name;
8   final String role;
9   final String photoURL;
10
11   /// * [name] berisi "nama user". Nilai defaultnya adalah "No Name".
12   ///
13   /// * [role] berisi peran/jabatan dari user. Nilai defaultnya adalah "No Role".
14   ///
15   /// * [photoURL] berisi link **foto user**. Nilai defaultnya adalah "null".
16   UserProfile(this.name = "No Name", this.role = "No Role", this.photoURL);
17
18   @override
19   Widget build(BuildContext context) {
20     return Column(
21       mainAxisAlignment: MainAxisAlignment.center,
22       crossAxisAlignment: CrossAxisAlignment.center,
23       children: <Widget>[
24         Image(
25           image: NetworkImage((photoURL != null)
26             ? photoURL
27             : "https://cdn.iconfinder.com/data/icons/web-ui-color/128/Account-512.png"), // NetworkImage
28           fit: BoxFit.cover,
29         ),
30       ],
31     );
32   }
33 }
```

Reloaded 2 of 463 libraries in 702ms.
Reloaded 2 of 463 libraries in 642ms.
Reloaded 2 of 463 libraries in 678ms.
Reloaded 2 of 463 libraries in 678ms.
Reloaded 2 of 463 libraries in 626ms.
Reloaded 2 of 463 libraries in 684ms.
Reloaded 2 of 463 libraries in 649ms.

```
1 import 'package:doc_comment/user_profile.dart';
2 import 'package:flutter/material.dart';
3
4 void main() => runApp(MyApp());
5
6 class MyApp extends StatelessWidget {
7   final UserProfile profile = UserProfile();
8
9   @override
10   Widget build(BuildContext context) {
11     return MaterialApp(
12       home: Scaffold(
13         appBar: AppBar(
14           backgroundColor: Colors.blue,
15           title: Text("Doc Com"),
16         ),
17         body: Center(child: profile),
18       ), // Scaffold
19     ); // MaterialApp
20   }
21 }
22 }
```

Reloaded 2 of 463 libraries in 702ms.
Reloaded 2 of 463 libraries in 642ms.
Reloaded 2 of 463 libraries in 678ms.
Reloaded 2 of 463 libraries in 678ms.
Reloaded 2 of 463 libraries in 626ms.
Reloaded 2 of 463 libraries in 684ms.
Reloaded 2 of 463 libraries in 649ms.

Kalau juga bisa kasih contoh kodingan di descripsi ini
Coba kita isi dahulu
Parameter nya
Misalkan seperti ini

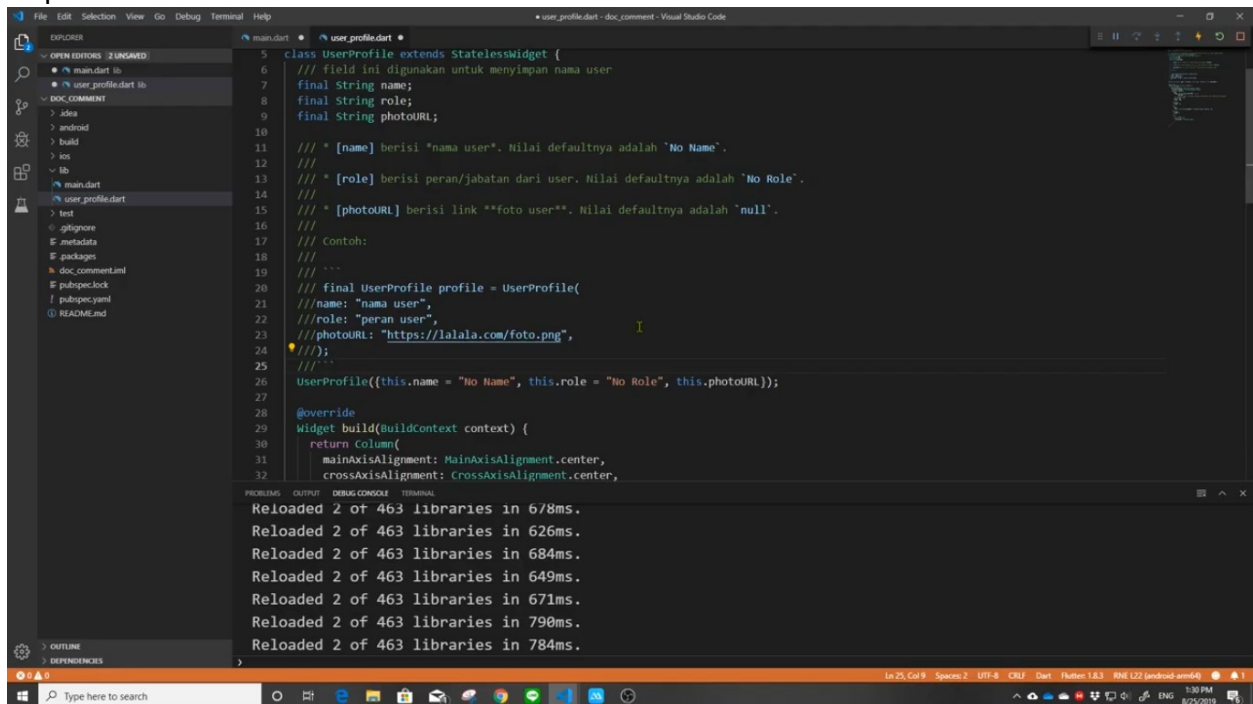


The screenshot shows a Visual Studio Code editor with a Flutter project. The Explorer pane on the left shows the project structure, including files like main.dart, user_profile.dart, and doc_comment.dart. The main editor displays the code for user_profile.dart, which defines a UserProfile class and a MyApp widget. The MyApp widget uses a Scaffold with an AppBar and a Center widget containing a Text widget. The terminal at the bottom shows the output of the app, indicating that it has been reloaded successfully.

```
1 import 'package:doc_comment/user_profile.dart';
2 import 'package:flutter/material.dart';
3
4 void main() => runApp(MyApp());
5
6 class MyApp extends StatelessWidget {
7   final UserProfile profile = UserProfile(name: "nama user", role: "peran user", photoURL: "https://lalala.com/fot.jpg");
8
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      home: Scaffold(
13        appBar: AppBar(
14          backgroundColor: Colors.red[900],
15          title: Text("Doc Comment Example"),
16        ), // AppBar
17        body: Center(child: profile),
18      ), // Scaffold
19    ); // MaterialApp
20  }
21 }
22
```

Reloaded 2 of 463 libraries in 678ms.
Reloaded 2 of 463 libraries in 626ms.
Reloaded 2 of 463 libraries in 684ms.
Reloaded 2 of 463 libraries in 649ms.
Reloaded 2 of 463 libraries in 671ms.
Reloaded 2 of 463 libraries in 790ms.
Reloaded 2 of 463 libraries in 784ms.

Mka kita tulis
Contoh codingan
Maka gunakan /// 3 kali
Seperti ini



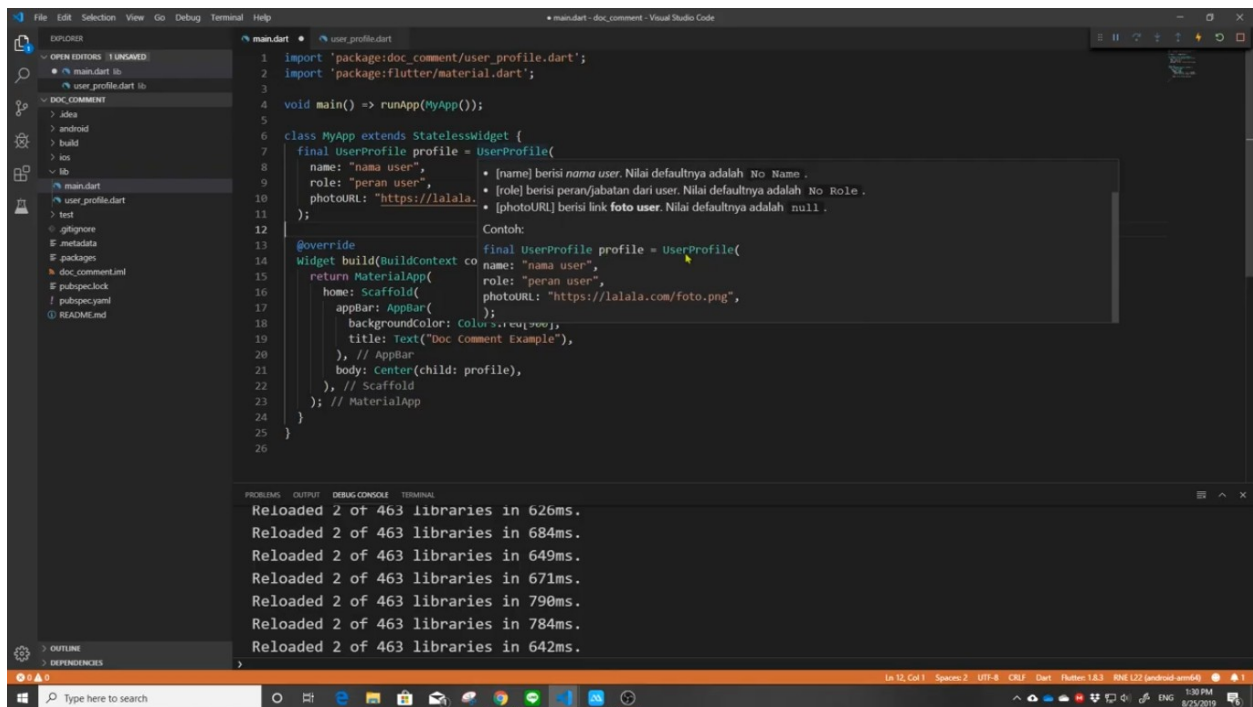
The screenshot shows a Visual Studio Code editor with a Flutter project. The Explorer pane on the left shows the project structure, including files like main.dart, user_profile.dart, and doc_comment.dart. The main editor displays the code for user_profile.dart, which defines a UserProfile class and a Column widget. The UserProfile class has three properties: name, role, and photoURL, each with a default value. The Column widget is used to display the UserProfile object. The terminal at the bottom shows the output of the app, indicating that it has been reloaded successfully.

```
5 class UserProfile extends StatelessWidget {
6   /// field ini digunakan untuk menyimpan nama user
7   final String name;
8   final String role;
9   final String photoURL;
10
11   /// * [name] berisi "nama user". Nilai defaultnya adalah "No Name".
12   ///
13   /// * [role] berisi peran/jabatan dari user. Nilai defaultnya adalah "No Role".
14   ///
15   /// * [photoURL] berisi link **foto user**. Nilai defaultnya adalah "null".
16   ///
17   /// Contoh:
18   ///
19   /// ```
20   /// final UserProfile profile = UserProfile(
21   ///   name: "nama user",
22   ///   role: "peran user",
23   ///   photoURL: "https://lalala.com/foto.png",
24   /// );
25   /// ```
26   UserProfile({this.name = "No Name", this.role = "No Role", this.photoURL});
27
28   @override
29   Widget build(BuildContext context) {
30     return Column(
31       mainAxisAlignment: MainAxisAlignment.center,
32       crossAxisAlignment: CrossAxisAlignment.center,

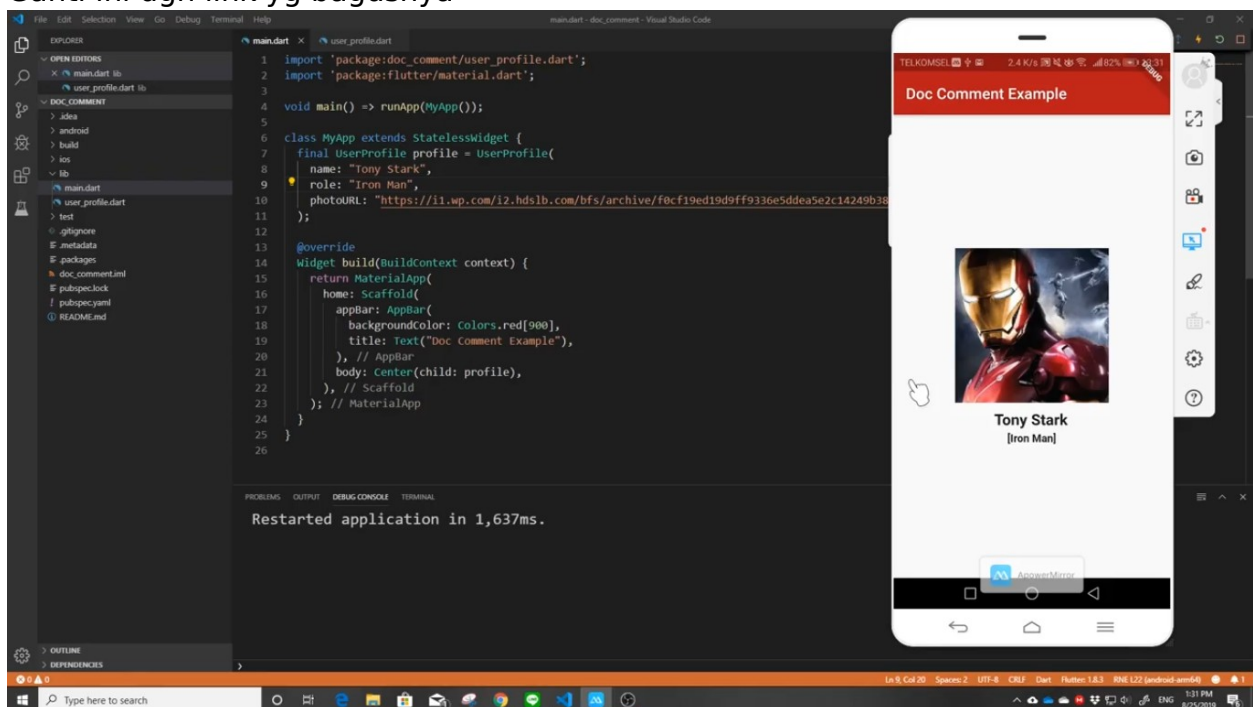
```

Reloaded 2 of 463 libraries in 678ms.
Reloaded 2 of 463 libraries in 626ms.
Reloaded 2 of 463 libraries in 684ms.
Reloaded 2 of 463 libraries in 649ms.
Reloaded 2 of 463 libraries in 671ms.
Reloaded 2 of 463 libraries in 790ms.
Reloaded 2 of 463 libraries in 784ms.

Dan bisa kita lihat



Ganti ini dgn link yg bagusnya



Maka hasilnya seperti ini

Jadi ini cara nya buat doc comment


```
5 class UserProfile extends StatelessWidget {
6   /// field ini digunakan untuk menyimpan nama user
7   final String name;
8   final String role;
9   final String photoURL;
10
11   /// * [name] berisi "nama user". Nilai defaultnya adalah "No Name".
12   ///
13   /// * [role] berisi peran/jabatan dari user. Nilai defaultnya adalah "No Role".
14   ///
15   /// * [photoURL] berisi link **foto user**, Nilai defaultnya adalah "null".
16   ///
17   /// Contoh:
18   ///
19   /// ```
20   /// final UserProfile profile = UserProfile(
21   ///   name: "nama user",
22   ///   role: "peran user",
23   ///   photoURL: "https://lalala.com/foto.png",
24   /// );
25   /// ```
26   UserProfile({this.name = "No Name", this.role = "No Role", this.photoURL});
27
28   @override
29   Widget build(BuildContext context) {
30     return Column(
31       mainAxisAlignment: MainAxisAlignment.center,
32       crossAxisAlignment: CrossAxisAlignment.center,
```

Restarted application in 1,637ms.

Ini cara komentar yg baik bisa lihat documentasi nya

greet(name) {
 // Assume we have a valid name.
 print('Hi, \$name!');
}

greet(name) {
 /* Assume we have a valid name. */
 print('Hi, \$name!');
}

You can use a block comment (/* ... */) to temporarily comment out a section of code, but all other comments should use //.

Doc comments

Doc comments are especially handy because `dartdoc` parses them and generates beautiful doc pages from them. A doc comment is any comment that appears before a declaration and uses the special `///` syntax that `dartdoc` looks for.

DO use `///` doc comments to document members and types.

Lint rule `slash_for_doc_comments`

Using a doc comment instead of a regular comment enables `dartdoc` to find it and generate documentation for it.

/// The number of characters in this chunk when unsplit.
int get length => ...

// The number of characters in this chunk when unsplit.
int get length => ...

For historical reasons, `dartdoc` supports two syntaxes of doc comments: `///` ('C# style') and `/** ... */` ('JavaDoc style'). We prefer `///` because it's more compact. `/** ... */` add two content-free lines to a multiline doc comment. The `///` syntax is also easier to read in some situations, such as when a doc comment contains a bulleted list that uses `*` to mark list items.

Contents

- Comments
- DO format comments like sentences
- DON'T use block comments for documentation
- Doc comments
- DO use `///` doc comments to document members and types
- PREFER writing doc comments for public APIs
- CONSIDER writing a library-level doc comment
- CONSIDER writing doc comments for private APIs
- DO start doc comments with a single-sentence summary
- DO separate the first sentence of a doc comment into its own paragraph
- Avoid redundancy with the surrounding context
- PREFER starting function or method comments with third-person verbs
- PREFER starting variable, getter or setter comments with

Maka ada komentar nya yg baik warna hijau jelek warna merah
Yg penting komunikasi antar tim agar tim mengerti
Agar gk bingung